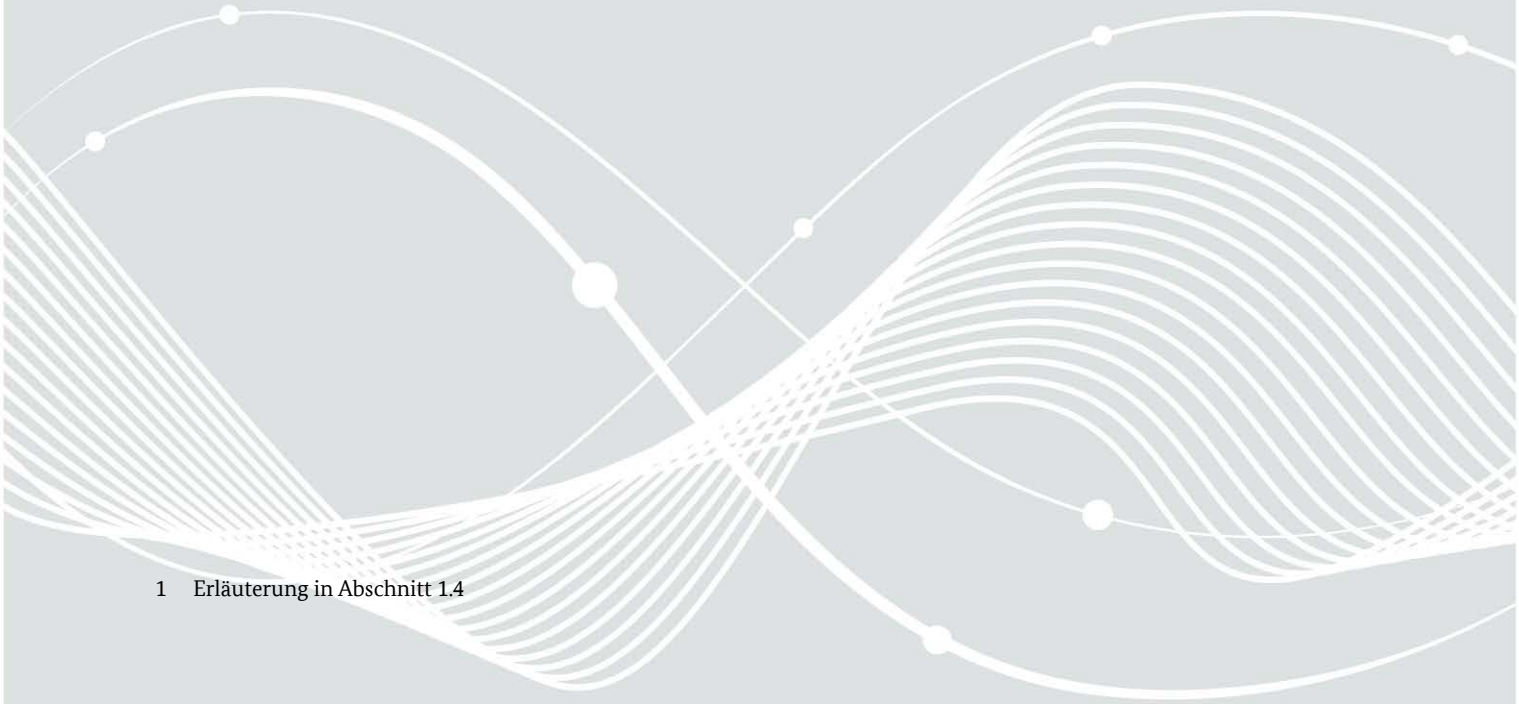
Bundesamt
für Sicherheit in der
Informationstechnik

Sicherheitsanforderungen an digitale Gesundheitsanwendungen

Technische Richtlinie BSI TR-03161

Trial Use[1]



1 Erläuterung in Abschnitt 1.4

Änderungshistorie

Version	Datum	Name	Beschreibung
1.0	15.04.2020	Kleinmanns (DI 24)	Erste Version

Bundesamt für Sicherheit in der Informationstechnik
Postfach 20 03 63
53133 Bonn
E-Mail: referat-di24@bsi.bund.de
Internet: https://www.bsi.bund.de

Inhaltsverzeichnis

1 Einleitung

1.1 Gegenstand der Technischen Richtlinie

Nach § 33a Sozialgesetzbuch Fünf (SGB V) haben gesetzlich Krankenversicherte unter bestimmten Voraussetzungen einen Anspruch auf Versorgung mit sogenannten digitalen Gesundheitsanwendungen. Sie sind dazu bestimmt, die „Erkennung, Überwachung, Behandlung oder Linderung von Krankheiten oder die Erkennung, Behandlung, Linderung oder Kompensierung von Verletzungen oder Behinderungen zu unterstützen“ [SGBV33a]. Diese Technische Richtlinie wendet sich an Hersteller von digitalen Gesundheitsanwendungen für mobile Endgeräte. Darüber hinaus kann sie als Richtlinie für mobile Anwendungen betrachtet werden, welche sensible Daten verarbeiten und speichern.

1.2 Zielsetzung der Technischen Richtlinie

Die Digitalisierung aller Lebensbereiche, sei es im Beruf, in Heimumgebungen, im Individual- oder im öffentlichen Personenverkehr, schreitet stetig voran. Bereits im Jahr 2018 überschritt die Anzahl der Internetnutzer die Grenze von vier Milliarden Menschen. Zwei Drittel der zur Zeit 7,6 Milliarden Menschen zählenden Weltbevölkerung nutzen ein Mobiltelefon. Mehr als drei Milliarden Menschen nutzen soziale Netzwerke und tun dies in neun von zehn Fällen über ihr Smartphone (vgl. [GDR18]). Diese Entwicklung setzt sich im Gesundheitswesen mit dem Trend zum „Self-Tracking“, aber auch mit der zunehmenden Forderung nach der effizienten Nutzung einmal erhobener medizinischer Daten, fort. Insbesondere im Gesundheitswesen ist es dabei komfortabel, dass orts- und zeitunabhängig auf die eigenen medizinischen Daten zugegriffen werden kann. Mobile Anwendungen speichern in diesem Fall sensible und persönliche Daten, von der Pulsfrequenz, über Aufzeichnungen des Schlafrhythmus bis hin zu Medikationsplänen sowie ärztlichen Verordnungen und Bescheinigungen. Sie verbinden den Nutzer mit entsprechenden Services und fungieren als Kommunikations-Knotenpunkte. Ein kompromittiertes Smartphone kann somit das gesamte digitale Leben des Nutzers ungewollt offenlegen. Das Einhalten von geeigneten Sicherheitsstandards, gerade im Bereich der mobilen Anwendungen, kann dies wesentlich erschweren und möglicherweise sogar verhindern. Schon während der Entwicklungsphase sollten Hersteller sehr verantwortungsvoll planen, wie eine mobile Anwendung personenbezogene und andere sensible Daten verarbeitet, speichert und schützt.

Die IT-Sicherheit verfolgt im wesentlichen drei Schutzziele: Vertraulichkeit, Integrität und Verfügbarkeit.

Gerade bei mobilen Gesundheitsanwendungen ist die Einhaltung dieser Anforderungen von besonderer Wichtigkeit. Im Gegensatz zum Finanzwesen, wo betrügerisch abgeführte Geldbeträge wieder von den Kreditinstituten an den Kunden rückerstattet werden können, ist die Vertraulichkeit von Gesundheitsdaten, die unwillentlich öffentlich gemacht werden, für immer verloren. Der Patient könnte hierfür zwar Schadensersatz erhalten, die Veröffentlichung kann allerdings nicht ungeschehen gemacht werden. Darüber hinaus können durch das ungewollte Bekanntwerden von Gesundheitsdaten, im sozialen, wie auch im beruflichen Umfeld, unerwünschte Folgen mit erheblichen Auswirkungen entstehen.

Sollte ein Angreifer in der Lage sein, Gesundheitsdaten eines Dritten zu manipulieren und damit deren Integrität zu verletzen, könnte er wesentlichen Einfluss auf Therapieentscheidungen und letztlich die Gesundheit des Betroffenen haben.

Diese Technische Richtlinie soll als Leitfaden dienen, um Entwickler von mobilen Anwendungen im Gesundheitswesen bei der Erstellung sicherer mobiler Applikationen zu unterstützen.

1.3 Übersicht der Technischen Richtlinie

1.3.1 Methodik

Im Folgenden bezeichnet der Begriff *Anwendung* eine mobile digitale Gesundheitsanwendung im Sinne des § 33a SGB V (siehe [SGBV33a]). Der Betrieb kann autonom mit einer *Applikation* auf dem Endgerät oder in Kombination mit einem sicheren *Backend* betrieben werden. Wird im Folgenden der Begriff *Backend* verwendet, ist insbesondere auch der Einsatz von Cloud Computing gemeint. Auf Grund des rasanten technischen Fortschritts und der Diversität mobiler Endgeräte sowie ihrer Plattformen, erhebt die Technische Richtlinie keinen Anspruch auf Vollständigkeit. Sie kann als Mindestanforderung für den sicheren Betrieb einer Anwendung betrachtet werden.

Die Technische Richtlinie formuliert eine Security Problem Definition (SPD), welche potenzielle Bedrohungsszenarien aufweist. Aus der SPD werden Prüfaspekte für mobile Anwendungen und deren Plattformen bzw. Einsatzumgebungen abgeleitet, um vor Bedrohungen zu schützen.

Die in dieser Technischen Richtlinie formulierten Bedrohungsszenarien und Prüfaspekte basieren auf Erfahrungen, die das BSI bei bisherigen Untersuchungen von mobilen Anwendungen im Gesundheitswesen gesammelt hat. Darüber hinaus orientiert sie sich an internationalen Standards, wie den „Smartphone Secure Development Guidelines“ [SSDG] und dem „Mobile AppSec Verification Standard“ [MASVS], mit seinem dazugehörigen „Mobile Security Testing Guide“ [MSTG].

Eine Grundanforderung an digitale Gesundheitsanwendungen ist die Orientierung an Best-Practice-Empfehlungen und anderen allgemeinen Anforderungen an sichere, verteilte Anwendungen. Dazu zählen die Durchführung intensiver funktionaler Tests, Integrationstests sowie insbesondere Positiv-/Negativ-Tests von Sicherheitsleistungen der Anwendung. Die TR stellt darüber hinaus zusätzliche, spezifische Anforderungen.

1.3.2 Begriffe

Diese Technische Richtlinie verwendet folgende Begriffe:

MUSS	Der Hersteller muss eine bestimmte Eigenschaft zwingend implementieren.
DARF NICHT	Die Anwendung/die Applikation/das Backend darf eine bestimmte Eigenschaft unter keinen Umständen aufweisen.
SOLL	Die Anwendung/die Applikation/das Backend muss eine bestimmte Eigenschaft aufweisen, außer es wird dargelegt, dass durch ein Nicht-Umsetzen kein Risiko für den sicheren Betrieb besteht, bzw. eine Umsetzung, aufgrund von technischen Einschränkungen, derzeit nicht möglich ist.
KANN	Die Anwendung/die Applikation/das Backend kann eine bestimmte Eigenschaft aufweisen, wobei ein Umsetzen dieser Eigenschaft vom Lösungsanbieter anzuzeigen ist.

1.4 Offene Punkte

Die Technische Richtlinie befindet sich im Status "trial use". Das heißt, dass Sicherheitsziele festgelegt sind, aber noch Erfahrungen mit der Anwendung der Prüfaspekte gesammelt werden sollen. Während der "trial use"-Phase wird verstärkt auf Rückmeldungen aus der Industrie geachtet. Hierdurch kann es zu

Änderungen innerhalb der einzelnen Prüfaspekte kommen. Es ist auch denkbar, dass Prüfaspekte hinzugefügt werden oder wegfallen.

In zukünftigen Versionen werden Kapitel hinzugefügt, welche eine Prüfung und Zertifizierung nach dieser Technischen Richtlinie ermöglichen.

2 Überblick der Sicherheitsanforderungen an mobile Anwendungen

2.1 Anwendungskonzepte auf mobilen Endgeräten

Der Begriff „mobile Anwendung“ bezeichnet ein Programm, das auf einer mobilen Plattform ausgeführt wird. Grundsätzlich lassen sich solche Anwendungen in drei Kategorien unterteilen. Die erste Kategorie bilden die nativen Applikationen (Kapitel 2.1.1), welche direkt auf die Plattform, auf der sie ausgeführt werden, zugeschnitten sind, ab. Dem gegenüber stehen die Web-Anwendungen (Kapitel 2.1.2). Ihre Implementierung ist völlig unabhängig von der Plattform und sie laufen innerhalb des Web-Browsers des Endgeräts. In die dritte Kategorie fallen die hybriden Ansätze (Kapitel 2.1.3). Sie spiegeln alle möglichen Kombinationen aus nativen Applikationen und Web-Anwendungen wider.

Da Web-Anwendungen weit über den allgemeinen Einsatz auf mobilen Endgeräten hinaus gehen, liegt der Fokus dieser Publikation auf nativen Anwendungen und dem nativen Teil von hybriden Ansätzen. Für zusätzliche Hinweise zur sicheren Entwicklung und zum sicheren Betrieb von Web-Anwendungen empfiehlt das BSI, weiterführende Literatur zu Rate zu ziehen. Der „OWASP Application Security Verification Standard“ [ASVS] bietet hier beispielsweise einen guten Einstieg.

2.1.1 Native-Applikationen

Eine native Applikation ist passend auf eine Plattform und deren Betriebssystem zugeschnitten. Sie basiert auf den von der Plattform (beispielsweise Android oder iOS) bereitgestellten Programmierwerkzeugen (Software Development Kits - SDKs). Diese ermöglichen einen direkten Zugriff auf Gerätekomponenten, wie beispielsweise GPS, Kamera oder Mikrofon. Aufgrund Ihrer Nähe zum Betriebssystem können sie eine sehr gute Performanz, eine hohe Zuverlässigkeit und eine intuitive Bedienbarkeit erreichen. Die Applikationen werden beispielsweise über den plattformeigenen App-Store installiert und können oft auch offline betrieben werden.

Mit der Nähe zum Betriebssystem sind allerdings auch Nachteile verbunden. Änderungen am Betriebssystem, beispielsweise durch Updates, können dazu führen, dass Anpassungen an der Applikation vorgenommen werden müssen. Sollte dies nicht erfolgen, kann es zu Beeinträchtigungen der Funktionsfähigkeit der Applikation kommen. Darüber hinaus ist es nicht möglich sie auf anderen Betriebssystemen zu installieren. Soll die gleiche Applikation auf mehreren Betriebssystemen publiziert werden, so muss jeweils eine eigene Codebasis[2] existieren. Dies ist häufig mit einem hohen Aufwand und somit auch hohen Kosten verbunden.

2.1.2 Web-Anwendungen

Web-Anwendungen sind Webseiten, die so programmiert sind, dass sie wie eine native Applikation aussehen und sich vergleichbar verhalten. Im Gegensatz zu nativen Applikationen basieren sie nicht auf einem SDK der zugrunde liegenden Plattformen, sondern auf klassischen Programmierwerkzeugen der Web-Entwicklung. In den meisten Fällen kommen HTML5 und JavaScript zum Einsatz. Aus diesem Grund ist mit ihnen nur ein sehr eingeschränkter Zugriff auf Gerätekomponenten möglich. Ihr größter Vorteil

2 Es existieren auch *Cross-Platform*-Implementierungsansätze, welche die Entwicklung einer Anwendung für verschiedene Plattformen gleichzeitig unterstützen. Allerdings verschiebt sich dadurch die Abhängigkeit lediglich in diese sehr umfangreiche Middleware, die alle Zielplattformen abdecken muss.

besteht jedoch darin, dass sie unabhängig vom Betriebssystem sind. Da die Anwendungen innerhalb eines Web-Browsers laufen, können sie auf jeder Plattform gleichermaßen eingesetzt werden, ohne Anpassungen an der Codebasis vornehmen zu müssen.

2.1.3 Hybride Ansätze

Hybride Anwendungen verbinden sowohl die Vor-, als auch die Nachteile von nativen Applikationen und Web-Anwendungen. Mit Hilfe des SDKs wird eine Rahmen-Applikation geschaffen, welche alle Vor- und Nachteile von nativen Anwendungen aufweist. Sie kann auf Gerätekomponenten zugreifen und über einen App-Store bezogen werden, jedoch nicht auf anderen Plattformen installiert werden, ohne Anpassungen am Quellcode vorzunehmen. Darüber hinaus beinhalten die Rahmen-Applikationen einen eingebetteten Web-Browser (WebView genannt), mit dessen Hilfe Web-Anwendungen in native Applikationen eingebunden werden können. Dadurch ist es auch Web-Anwendungen möglich auf die sonst nur den nativen Applikationen vorbehaltenen Gerätekomponenten zuzugreifen. Allerdings müssen dabei Einbußen bei der Performanz und Stabilität in Kauf genommen werden. Darüber hinaus kann es durch den Einsatz unterschiedlicher Benutzerschnittstellen zu einem Rückgang der User-Experience kommen. Die Plattformabhängigkeit der Anwendung bezieht sich nun lediglich auf die Rahmen-Applikation, womit der Aufwand für eine Migration auf andere Plattformen deutlich reduziert wird.

2.2 Backend-Services

Die meisten Anwendungen verlassen sich für die Verarbeitung und Speicherung von Daten nicht ausschließlich auf die von der Laufzeitumgebung bereitgestellten Ressourcen. Sie lagern diese Aufgaben auf ein zentrales System im Hintergrund (Backend) aus. Neben der Verarbeitung und Speicherung von Daten übernehmen diese Systeme oft auch Aufgaben zur Authentifizierung und Autorisierung von Nutzern oder andere zentrale Tätigkeiten. Dies erlaubt es, dass nicht alle Funktionalitäten der Applikationen auf den Endgeräten umgesetzt werden müssen. Oft beschränken sie sich auf eine grafische Nutzerführung (Frontend). Eine generelle Aussage darüber, wie viel Funktionalität in der Applikation selbst umgesetzt und wie viel auf einen Server ausgelagert wird, kann nicht getroffen werden. Die Ausprägungen können von Anwendung zu Anwendung variieren.

Für die Nutzung von Anwendungen, die an ein Backend angeschlossen sind, ist eine aktive Internetverbindung zwingend erforderlich. Dabei wird für die Kommunikation zwischen Front- und Backend meist eine über HTTPS gesicherte Verbindung eingesetzt. Der Einsatz von Backend-Systemen beschränkt sich nicht nur auf den Bereich der mobilen Anwendungen, sondern spiegelt den aktuellen Stand der Technik für fast alle Anwendungen wieder. Da der Fokus dieser Publikation auf mobilen Anwendungen liegt, beziehen sich die nachfolgende Bedrohungsanalyse und die Prüfaspekte auf den Schutz der Applikation und zusätzlich lediglich auf jene Eigenschaften des Backends, welche direkten Einfluss auf die Sicherheit der Applikation haben. Für weiterführende Hinweise zum sicheren Betrieb und der sicheren Entwicklung von Backend-Systemen empfiehlt es sich weiterführende Literatur zu studieren. Einen guten Einstieg bieten die „Top 10 Web Application Security Risks“ der OWASP Foundation [T10WASR]. Beim Einsatz von Cloud Computing wird empfohlen den „Kriterienkatalog Cloud Computing“ des BSI [KCC-C5] zu verwenden.

2.3 Security Problem Definition

Die Security Problem Definition beschreibt Annahmen, Bedrohungen und organisatorische Sicherheitspolitiken, die für digitale Gesundheitsanwendungen zur Erbringung der Sicherheitsleistung relevant sind.

2.3.1 Annahmen

A.Device	Die Plattform, auf der die Applikation genutzt wird, wird vom Nutzer selbst betrieben und vor Schwachstellen geschützt, etwa durch Aktualisieren des Betriebssystems nach Bereitstellung der jeweiligen Updates. Ihre Sicherheit wurde nicht durch das vorsätzliche Durchführen von „Roots“ oder „Jailbreaks“[3] beeinträchtigt.

2.3.2 Bedrohungen

T.AssetLocal	Ein Unbefugter erhält Zugriff auf sensitive Daten der Applikation, etwa auf unverschlüsselt gespeicherte Daten im Dateisystem oder Arbeitsspeicher. Dies umfasst auch, dass ein Angreifer auf verschlüsselte, sensitive Daten, nach Analyse des Verschlüsselungsmechanismus, im Klartext zugreifen kann.
T.AssetRemote	Auf dem Backend kann unberechtigterweise auf Assets der Nutzer sowie der Anwendung (z. B. Entropie für operative TLS-Verbindung) zugegriffen werden. Dies erfolgt etwa über eine missbräuchliche Nutzung der mobilen Applikation oder über eine Schwachstelle in der Implementierung der Schnittstelle des Backends in Richtung Frontend.
T.Auth	Ein Angreifer erhält unter einer fremden Nutzerkennung oder der Verwendung fremder Rollen- oder Gruppenzugehörigkeit Zugriff auf sensible Daten anderer Nutzer.
T.Eavesdropping	Einem Angreifer gelingt es über eine unzureichend verschlüsselte/nicht-authentisierte Verbindung die Kommunikation der Anwendung zu belauschen oder zu beeinträchtigen. Es wird eine Transportverbindung zu einer unberechtigten Komponente aufgebaut, z. B. aufgrund mangelnder Prüfung von Zertifikatseigenschaften.
T.Expense	Die Anwendung verursacht unvorhergesehene, zusätzliche Kosten für den Nutzer.
T.Integrity	Ein Angreifer ist in der Lage Daten innerhalb eines Speichers oder über den Transportweg unbemerkt zu manipulieren.
T.Guessing	Ein Angreifer könnte ein Passwort durch Ausprobieren erraten und dadurch unberechtigten Zugriff auf sensible Daten eines anderen Nutzers erhalten.
T.MemoryStructures	Ein Angreifer führt ein Reverse Engineering auf die Applikation durch und ermittelt dadurch ungeschützte Datenstrukturen im Speicher, wodurch der Zugriff auf Schlüssel und sensible Daten möglich wird.

3 Das Aufweichen der betriebssystemeigenen Sicherheitsfunktionalitäten durch Erlangen von erhöhten Zugriffsrechten und Ermöglichen der Installation von Applikationen aus unbekannten Quellen.

2.3.3 Organisatorische Sicherheitspolitiken

OSP.Authorization	Der Hersteller entwickelt ein Autorisierungskonzept, welches sowohl den lesenden, als auch den schreibenden Zugriff auf sensible Daten steuert. Die Zugriffsberechtigungen müssen so gewählt werden, dass ausschließlich für die Erfüllung des primären Zwecks erforderliche Rechte erteilt werden. Das Autorisierungskonzept muss unabhängig von der Authentifizierung implementiert werden.
OSP.BackendConnLog	Auf dem Backend werden Informationen über alle ausgehenden Verbindungen zur Post-Mortem Analyse von Sicherheitsvorfällen erfasst, unter anderem mit Metainformationen über verwendete Proxies und überprüfte Server-Zertifikate.
OSP.CriticalUpdates	Der Hersteller überprüft und überwacht die Anwendung sowie die genutzten Frameworks und Bibliotheken[4] dauerhaft auf ausnutzbare Schwachstellen. Der Hersteller muss bei ausnutzbaren Schwachstellen kurzfristig ein Update bereitstellen. Das Backend muss die Applikation über das Update informieren und nach einer Übergangsfrist (Grace Period) die Benutzung der Anwendung mit einer veralteten Applikation unterbinden.
OSP.LibsIn	Von externen Bibliotheken eingehende Daten sollen vor einer Verwendung in der Anwendung validiert werden (z. B. XML-Schemavalidierung, Prüfung auf ungültiges Encoding, etc.). Ziel ist es, die Anwendung vor Angriffen durch bösartige Eingaben zu schützen.
OSP.LibsOut	Die Anwendung soll sensible Daten nicht im Klartext an externe Frameworks bzw. Bibliotheken weitergeben. Zulässig ist die Nutzung entsprechender Frameworks bzw. Bibliotheken für die Sicherung eines Kommunikationskanals oder eines lokalen Speichercontainers.
OSP.RNG	Zufallszahlen sind aus einem Zufallszahlengenerator zu beziehen, der eine hohe Entropie besitzt. Die Applikation soll initial einmalig Entropie vom Nutzer in den Zufallszahlengenerator der Plattform einbringen. Anschließend bezieht die Applikation Zufall vom Backend und bringt diese in den lokalen Zufallszahlengenerator ein.
OSP.Purpose	Jegliche Datenerhebung, -verarbeitung, -speicherung und -weitergabe darf nur mit einer Zweckbindung erfolgen. Der Hersteller veröffentlicht dafür den primären Zweck der Anwendung und darüber hinaus welche Daten wie verarbeitet werden und wo und wie lange sie gespeichert werden. Ausgehend vom primären Zweck ist das zulässige Kommunikationsverhalten sowie die verwendete interne und externe Sensorik auszuwählen. Anwendungshinweis/Beispiel: Ortungsdaten wie WiFi-SSID, GPS u. ä. dürfen nur zweckgebunden und datensparsam verwendet werden. Sie dürfen nicht direkt oder indirekt (etwa in Bildaufnahmen) im Gerät persistiert werden, sofern dies nicht unmittelbar durch den Verwendungszweck erforderlich ist.

4 Unter einem externen Framework bzw. einer externen Bibliothek soll die Zusammenfassung von Funktionalität verstanden werden, die nicht in der Hoheit des Entwicklers der Anwendung entstanden ist und die auch nicht Teil der Funktionalität der verwendeten Betriebssystemplattform ist.

2.3.4 Restrisiken

Der Betrieb digitaler Gesundheitsanwendungen hat besonders hohe Anforderungen, die mit bestehenden Endgeräten und Cloud-Lösungen nur unzureichend abzudecken sind. Daher weist die Technische Richtlinie auf bestehende Restrisiken hin:

Mobile Endgeräte sind besonders anfällig für Diebstahl. Die offene Architektur vieler Plattformen begünstigt den Einsatz von Malware. Installierte Apps können bestehende Schwachstellen ausnutzten. Eine besondere Herausforderung ist der Schutz von Informationen während der Verarbeitung im Hauptspeicher.

Der Betrieb des Backends bei Public Cloud-Anbietern beinhaltet besondere Risiken für die sensiblen Daten der Nutzer. Während hohe Entropie, sichere Kommunikations- und Verschlüsselungsverfahren Risiken abmildern, sind in der Cloud Daten während der Verarbeitung quasi ungeschützt. Dies stellt höchste Anforderungen an den Betreiber der Cloud, sowie an andere Anwender, die eventuell gleichzeitig Ressourcen der selben physischen Maschine benutzen dürfen. Durch Ausbrechen aus der Virtuellen Maschine erhält ein Angreifer Zugriffsmöglichkeiten außerhalb seines Mandantenbereichs und kann unter Umständen sensible Daten eines anderen Mandanten (hier: der digitalen Gesundheitsanwendung), während deren Verarbeitung, einsehen und manipulieren.

Der Schutz von Kommunikationsverbindungen zwischen der Plattform, der Applikation und dem Backend erfolgt mittels des kryptographisch gesicherten TLS-Protokolls. Im vorliegenden Szenario geht die TR von einer einseitigen Authentisierung aus, wobei die Applikation die Authentizität des Backends prüft. Die Applikation fügt in den Prozess des TLS-Verbindungsaufbaus eigenen Zufall ein, um damit zu erschweren, dass ein Angreifer in die TLS-Verbindung eindringt. Zufallszahlen auf Smartphone-Plattformen erreichen im Allgemeinen jedoch nicht die notwendige Qualität, die für den Schutz sensibler Daten innerhalb einer digitalen Gesundheitsanwendung notwendig sind. Das Restrisiko während des Verbindungsaufbaus besteht darin, dass der Angreifer die Authentizität eigener Nachrichten vortäuschen kann. Dadurch könnte der Angreifer sensible Daten, welche von der Anwendung an das Backend übermittelt werden, einsehen und manipulieren. Der Anwendungshinweis zu O.Random_4 sieht eine Gegenmaßnahme vor, die dieses Restrisiko für die zweite, mit neuer Entropie angereicherte TLS-Verbindung, senken kann.

Im Allgemeinen ist auf Grund der in Kapitel 2.1 und 2.2 beschriebenen Einschränkungen, bezogen auf den Umfang der Technischen Richtlinie, eine gesamtheitliche Aussage über die Sicherheit der Anwendung, selbst unter Berücksichtigung aller aufgeführten Prüfaspekte, nicht möglich. Um die Sicherheit der gesamten Anwendung zu erhöhen, ist es erforderlich weitere Literatur zu studieren. Dies gilt insbesondere für den Schutz vor Angriffen, welche direkt das eingesetzte Backend als Ziel haben und bei der Verbindung von digitalen Gesundheitsanwendungen mit IoT-Geräten.

3 Prüfaspekte einer digitalen Gesundheitsanwendung nach Technischer Richtlinie

3.1 Prüfaspekte

Die Prüfung nach der Technischen Richtlinie deckt die minimalen Sicherheitseigenschaften digitaler Gesundheitsanwendungen ab. Die zu prüfende Sicherheitsfunktionalität lässt sich in folgende Prüfaspekte gliedern:

(1) Prüfung des Anwendungszwecks

(2) Prüfung der Architektur

(3) Prüfung des Quellcodes

(4) Prüfung der Drittanbieter-Software

(5) Prüfung der kryptographischen Umsetzung

(6) Prüfung der Authentifizierung

(7) Prüfung der Datenspeicherung und des Datenschutzes

(8) Prüfung der kostenpflichtigen Ressourcen

(9) Prüfung der plattformspezifischen Interaktionen

(10) Prüfung der Netzwerkkommunikation

(11) Prüfung der Resilienz

Der Hersteller dokumentiert für jeden Prüfaspekt, sofern die zu schützende Funktionalität verwendet wird, wie dessen Anforderung durch die Implementierung sichergestellt wird.

3.1.1 Prüfaspekt (1): Anwendungszweck

O.Zweck_1	Der Hersteller MUSS den primären Zweck der Anwendung und die Verwendung von personenbezogenen Daten vor der Installation offenlegen (etwa in der Beschreibung des App-Stores) und den Nutzer mindestens bei der erstmaligen Inbetriebnahme darüber informieren.
O.Zweck_2	Die Anwendung DARF Daten NICHT erheben und verarbeiten, die nicht dem primären Zweck der Anwendung dienen.
O.Zweck_3	Die Anwendung MUSS vor jeglicher Erfassung oder Verarbeitung personenbezogener Daten eine Einwilligungserklärung des Nutzers einholen.
O.Zweck_4	Daten, deren Verwendung der Benutzer nicht ausdrücklich zugestimmt hat, DÜRFEN NICHT von der Applikation oder dem Backend genutzt werden.

O.Zweck_5	Die Anwendung MUSS ermöglichen, dass der Nutzer seine Einwilligung wieder entziehen kann. Sie MUSS darüber informieren, inwieweit sich das Verhalten der Anwendung dadurch verändert.
O.Zweck_6	Der Hersteller MUSS ein Verzeichnis führen, welches erkennen lässt, welche Nutzereinwilligungen vorliegen. Der nutzerspezifische Teil des Verzeichnisses MUSS für den Nutzer einsehbar sein.
O.Zweck_7	Setzt die Anwendung Frameworks und Bibliotheken von Dritten ein, SOLLEN alle verwendeten Funktionen für den primären Zweck der Anwendung erforderlich sein. Die Anwendung SOLL anderweitige Funktionen sicher deaktivieren. Anwendungshinweis/Beispiel: Eine API für soziale Netzwerke dürfte nur verwendet werden, wenn dies mit dem primären Zweck der Anwendung vereinbar ist.
O.Zweck_8	Sofern es nicht für den vorgesehenen primären Zweck einer Anwendung erforderlich ist, DÜRFEN sensible Daten NICHT mit Dritten geteilt werden. Die Applikation MUSS den Nutzer über die Konsequenzen einer eventuellen Weitergabe der Daten vollumfänglich informieren und sein Einverständnis einholen (OPT-IN). Anwendungshinweis/Beispiel: Nutzt die Anwendung eine Kartenvisualisierung eines Drittherstellers, muss der Nutzer darauf hingewiesen werden, dass möglicherweise bestimmte Daten an Dritte abfließen.
O.Zweck_9	Die Anwendung DARF sensible Daten NICHT auf dem Bildschirm darstellen, außer dies ist für den Zweck der Anwendung erforderlich.

3.1.2 Prüfaspekt (2): Architektur

O.Arch_1	Security MUSS ein fester Bestandteil des Softwareentwicklungs- und Lebenszyklus‘ für die gesamte Anwendung sein (vgl. iOS Security Framework [iOSSF], beziehungsweise Security for Android Developers [SfAD]).
O.Arch_2	Bereits in der Design-Phase der Applikation MUSS berücksichtigt werden, dass die Applikation in der Produktiv-Phase sensible Daten verarbeiten wird. Die Architektur der Anwendung MUSS dafür die sichere Erhebung, Verarbeitung, Speicherung und Löschung der sensiblen Daten in einem Datenlebenszyklus abbilden.
O.Arch_3	Der Lebenszyklus von kryptographischem Schlüsselmaterial MUSS einer ausgearbeiteten Richtlinie folgen, die Eigenschaften wie die Zufallszahlenquelle, detaillierte Angaben zur Aufgabentrennung von Schlüsseln, Ablauf von Schlüsselzertifikaten, Integritätssicherung durch Hash-Algorithmen, etc., umfasst. Die Richtlinie SOLL auf anerkannten Standards wie [TR02102-1] und [NSP80057] basieren.
O.Arch_4	Verwendet die Anwendung eine Cloud als Backend-System, MUSS die Cloud über ein C5 (Cloud Computing Compliance Controls Catalogue) [KCC-C5] oder ein vergleichbares Zertifikat verfügen.

O.Arch_5	Vom Betriebssystem gesteuerte Backups und Cloud-Backups DÜRFEN KEINE unverschlüsselten sensiblen Daten und insbesondere kein Schlüsselmaterial beinhalten.
O.Arch_6	Sicherheitsfunktionen MÜSSEN immer, sowohl in der Applikation und im Backend, als auch auf allen Außenschnittstellen und API-Endpunkten, implementiert werden.
O.Arch_7	Die Anwendung MUSS einen Authentizitäts- und Integritätsschutz für die Applikation und ihre Konfiguration gewährleisten. Die Applikation SOLL dabei regelmäßig eine eigene Authentizitäts- und Integritätsprüfung des Applikations-Binaries, basierend auf einer digitalen Signatur mit Zertifikat, durchführen.
O.Arch_8	Nutzt die Anwendung Frameworks oder Bibliotheken von Dritten (etwa für Objektserialisierung), MUSS der Hersteller dem Nutzer Informationen über den Nutzungsumfang und die eingesetzten Sicherheitsmechanismen klar darstellen. Die Anwendung MUSS sicherstellen, dass diese Funktionen in sicherer Weise genutzt werden. Die Anwendung MUSS darüber hinaus sicherstellen, dass ungenutzte Funktionen durch Dritte nicht aktiviert werden können.
O.Arch_9	Interpretierter Code[5], der in möglicher Interaktionen mit Benutzereingaben steht (Webviews mit JavaScript), DARF KEINEN Zugriff auf verschlüsselte Speicher oder Nutzerdaten haben, sofern es für die Erfüllung des Zwecks der Anwendung nicht zwingend erforderlich ist.
O.Arch_10	Der Hersteller MUSS dem Nutzer eine barrierearme Möglichkeit bereitstellen, um Sicherheitsprobleme zu melden.
O.Arch_11	Das Backend SOLL sicherheitsrelevante Updates der Applikation erzwingen können.
O.Arch_12	Der Hersteller KANN die Applikation und Updates über einen vertrauenswürdigen Kanal mit einem eigenen App-Store bereitstellen.
O.Arch_13	Werden die Applikation und Updates nicht über die normalen Mechanismen des App-Stores der Hardwareplattform eingespielt, MÜSSEN sie durch Einsatz kryptographischer Maßnahmen verschlüsselt und signiert werden.

3.1.3 Prüfaspekt (3): Quellcode

O.Source_1	Nutzereingaben MÜSSEN vor deren Verwendung geprüft werden, um potenziell bösartige Werte vor der Verarbeitung herauszufiltern.
O.Source_2	Der Hersteller MUSS strukturierte Daten mit einer Escape-Syntax versehen.
O.Source_3	Fehlermeldungen und Benachrichtigungen DÜRFEN KEINE sensiblen Daten (z. B. user identifier) enthalten.
O.Source_4	Potenzielle Ausnahmen im Programmablauf (Exceptions) MÜSSEN abgefangen, kontrolliert behandelt und dokumentiert werden.

5 Nicht gemeint ist Code der plattformspezifischen Programmiersprachen (z. B. Java oder Kotlin bei Android).

O.Source_5	Bei Ausnahmen im Programmablauf (Exceptions), mit sicherheitskritischen Auswirkungen, SOLL die Anwendung Zugriffe auf sensible Daten abbrechen.
O.Source_6	Bei Programmumgebungen mit manueller Speicherverwaltung (d.h., die Applikation kann selbst exakt festlegen, wann und wo Speicher gelesen und beschrieben wird) MÜSSEN Applikation und Backend-Implementierung für lesende und schreibende Zugriffe auf Speichersegmente auf sichere Funktionsalternativen (z. B. sprintf_s statt printf) zurückgreifen.
O.Source_7	Alle Optionen zur Unterstützung der Entwicklung (z. B. Log-Aufrufe, Entwickler-URLs, Testmethoden, etc.) MÜSSEN in der Produktiv-Version deaktiviert sein.
O.Source_8	Der Hersteller MUSS sicherstellen, dass keine Debug-Mechanismen in der Produktiv-Version verbleiben.
O.Source_9	Die Implementierung der Anwendung SOLL moderne Sicherheitsmechanismen der Entwicklungsumgebung, wie beispielsweise Byte-Code Minimierung und Stack-Protection, aktivieren.

3.1.4 Prüfaspekt (4): Drittanbieter-Software

O.TrdP_1	Externe Bibliotheken und Frameworks SOLLEN in der aktuellsten verfügbaren Version, bezogen auf das genutzte Betriebssystem der Plattform, verwendet werden.
O.TrdP_2	Externe Bibliotheken und Frameworks MÜSSEN durch den Hersteller regelmäßig auf Schwachstellen überprüft werden. Funktionen aus Bibliotheken und Frameworks DÜRFEN bei bekannten Schwachstellen NICHT eingesetzt werden.
O.TrdP_3	Sicherheitsupdates für externe Bibliotheken und Frameworks MÜSSEN zeitnah eingespielt werden. Der Hersteller MUSS ein Sicherheitskonzept vorlegen, dass anhand der Kritikalität ausnutzbarer Schwachstellen die geduldete Weiternutzung für die Applikation, bzw. das Backend festlegt. Nachdem die Übergangsfrist (Grace Period) abgelaufen ist, MUSS die Anwendung den Betrieb verweigern.
O.TrdP_4	Der Nutzer MUSS über Mitigationsmaßnahmen informiert werden, sofern diese durch den Nutzer umsetzbar sind.
O.TrdP_5	Vor der Verwendung von externen Bibliotheken und Frameworks MUSS deren Quelle auf Vertrauenswürdigkeit geprüft werden.
O.TrdP_6	Die Anwendung SOLL sensible Daten nicht an Drittanbieter-Software weitergeben.
O.TrdP_7	Über Drittanbieter-Software eingehende Daten SOLLEN validiert werden. Anwendungshinweis/Beispiel: Ausnahmen zu O.TrdP_6 und O.TrdP_7 bilden beispielsweise Frameworks und Bibliotheken zur Verschlüsselung (TLS).
O.TrdP_8	Drittanbieter-Software, die nicht länger vom Hersteller oder Entwickler gewartet wird, DARF NICHT verwendet werden.

3.1.5 Prüfaspekt (5): Kryptographische Umsetzung

O.Cryp_1	Beim Einsatz von Verschlüsselung in der Applikation DÜRFEN KEINE fest einprogrammierten Schlüssel eingesetzt werden. Ausgenommen sind Techniken, die den verwendeten Schlüssel stark vor Reverse Engineering nach aktuellem Stand der Technik verbergen (Stichwort: „White Box Cryptography“). Werden statische Schlüssel eingesetzt, MUSS in einer mehrschichtigen Verschlüsselung mindestens ein nicht-statischer Schlüssel eingesetzt werden.
O.Cryp_2	Die Anwendung MUSS auf bewährte Implementierungen zur Umsetzung kryptographischer Primitive zurückgreifen (vgl. [TR02102-1]).
O.Cryp_3	Die Wahl kryptographischer Primitive MUSS passend zum Anwendungsfall sein und den Vorgaben des aktuellen Stands der Technik (siehe [TR02102-1]) entsprechen.
O.Cryp_4	Kryptographische Schlüssel DÜRFEN NICHT für mehr als genau einen Zweck eingesetzt werden. Anwendungshinweis/Beispiel: Es wird der Zweck nach Schutz durch Verschlüsselung und Authentisierung unterschieden. Dafür müssen unterschiedliche Schlüssel vorgesehen werden.
O.Cryp_5	Die Stärke der kryptographischen Schlüssel MUSS dem aktuellen Stand der Technik entsprechen. (siehe [TR02102-1])
O.Cryp_6	Alle kryptographischen Schlüssel SOLLEN in einer vor Manipulation geschützten Umgebung liegen (z. B. embedded Secure Element/Trusted Execution Environment). Dabei sind Varianten für verschiedene Hardware-Plattformen zu berücksichtigen.

3.1.5.1 Zufallszahlen

O.Random_1	Alle Zufallswerte MÜSSEN über einen sicheren kryptographischen Zufallszahlengenerator erzeugt werden.
O.Random_2	Die Anwendung MUSS Zufallszahlen von einem Zufallszahlengenerator mit hoher Entropie beziehen.
O.Random_3	Die Anwendung SOLL dem Zufallszahlengenerator einen Startwert (Seed) zuweisen, der sich aus mindestens drei voneinander unabhängigen Systemparametern zusammensetzt. Die Parameter SOLLEN von außerhalb der Anwendung nicht ermittelbar sein. Stellt die Plattform Hardware-Zufallszahlengeneratoren zur Verfügung, welche keine Vergabe von Startwerten erlauben KÖNNEN stattdessen diese verwendet werden. Anwendungshinweis/Beispiel: Dies betrifft die Zufallszahlengeneratoren auf dem Endgerät der Applikation sowie die des Backends.

O.Random_4	Die Applikation SOLL bei Erstellung eines Startwerts (Seed) für den Zufallszahlengenerator einen geeigneten Zufall vom Backend beziehen. Anwendungshinweis/Beispiel: Die Applikation bringt vor der erstmaligen TLS-Verbindung Entropie, gemäß O.Random_3 (z. B. aus Benutzerinteraktion und Gerätesensorik), durch einen Seed in den lokalen Zufallszahlengenerator ein. Sie baut eine initiale Verbindung zum Erhalt zusätzlicher Entropie aus der Zufallszahlenquelle des Backends auf. Die Verbindung wird anschließend sofort wieder abgebaut. Die Applikation berücksichtigt den erhaltenen Zufall, entsprechend O.Random_4, im lokalen Zufallszahlengenerator. Sie verwendet für die operationelle TLS-Verbindung von nun an Zufall aus der lokalen Zufallsquelle, welche mit der Entropie der Zufallszahlenquelle des Backends angereichert wurde.

3.1.6 Prüfaspekt (6): Authentifizierung

O.Auth_1	Der Hersteller MUSS ein Konzept zur Authentifizierung (zwei Faktoren), Autorisierung (Rollenkonzept) und zum Beenden einer Anwendungssitzung dokumentieren.
O.Auth_2	Für die Anbindung eines Backend-Systems MUSS an der Backend-Schnittstelle eine geeignete Authentifizierung und Autorisierung stattfinden.
O.Auth_3	Die Applikation SOLL Authentifizierungsmechanismen und Autorisierungsfunktionen separat realisieren. Sind für die Anwendung verschiedene Rollen notwendig, MUSS eine Autorisierung bei jedem Datenzugriff separat realisiert werden.
O.Auth_4	Der Nutzer MUSS mittels zweitem Faktor authentifiziert werden, bevor sensible Daten in der Anwendung verarbeitet werden (Step-Up Authentisierung).
O.Auth_5	Für die Nutzer-Authentifizierung in der Anwendungssitzung KANN der zweite Faktor vom Backend-System erzeugt werden.
O.Auth_6	Für die Bewertung eines Authentifizierungsvorgangs SOLLEN zusätzliche Informationen (z. B. das verwendete Endgerät, verwendeter WiFi-Zugangsknoten oder Zeit des Zugriffs) mit einbezogen werden. Bei einer Abweichung von gewohnten Parametern MUSS eine zusätzliche Authentifizierungsmaßnahme (Step-Up Authentisierung) erfolgen.
O.Auth_7	Bei einer Authentifizierung mittels Benutzername und Passwort MÜSSEN starke Passwortrichtlinien existieren.
O.Auth_8	Für die Authentifizierung mittels Benutzername und Passwort KANN die Stärke des verwendeten Passworts dem Nutzer angezeigt werden. Informationen über die Stärke des gewählten Passworts DÜRFEN NICHT im Applikationsspeicher oder im Backend gehalten werden.
O.Auth_9	Der Nutzer MUSS die Möglichkeit haben sein Passwort zu ändern.
O.Auth_10	Backend und Applikation MÜSSEN Maßnahmen vorsehen, die ein Ausprobieren von Login-Parametern (z. B. Passwörter) verhindern. Dies kann beispielsweise

durch Verzögerung nachfolgender Login-Versuche oder den Einsatz von sogenannten Captchas erreicht werden.

O.Auth_11 Wurde die Anwendung unterbrochen (in den Hintergrundmodus versetzt) MUSS eine erneute Authentifizierung gefordert werden.

3.1.6.1 Authentifizierung über Biometrie

O.Biom_1 Die Verwendung biometrischer Sensoren SOLL nicht als alleiniger Authentifizierungsmechanismus eingesetzt werden. Sie ist lediglich als Teil einer Zwei-Faktor-Authentifizierung zulässig.

O.Biom_2 Der Hersteller MUSS definieren, welche Qualität und Eigenschaften ein biometrischer Sensor mindestens aufweisen muss, um von der Anwendung verwendet werden zu dürfen.

O.Biom_3 Die Applikation MUSS die Hardware des biometrischen Sensors, vor dem Einsatz, über eine White- oder eine Blacklist prüfen. Der Hersteller SOLL eine White-/Blacklist mit sicherer/unsicherer Biometrie-Sensorik pflegen. Eine White-/Blacklist MUSS im Backend aktuell verfügbar gehalten werden.

O.Biom_4 Vor jeder Authentifizierung über einen biometrischen Sensor MUSS die Applikation sicherstellen, dass die zur Verfügung stehende Hardware den festgelegten Anforderung genügt.

O.Biom_5 Bevor die Applikation einen biometrischen Sensor nutzt, MUSS diese sicherstellen, dass dem Sensor biometrische Referenzmerkmale des Gerätenutzers zum Vergleich bereit stehen.

O.Biom_6 Die Applikation MUSS feststellen, wann die biometrischen Referenzmerkmale verändert wurden und die Anmeldung ablehnen, falls biometrische Referenzmerkmale nachträglich (das heißt seit der Aktivierung des Authentifizierungskontrollmechanismus in der Applikation) verändert worden sind.

O.Biom_7 Die Applikation MUSS zur Auswertung der biometrischen Authentifizierung auf betriebssystemeigene Funktionalitäten zurückgreifen (z. B. Entsperren KeyChain/ KeyStore).

3.1.6.2 Zustandsbezogene Authentifizierungsmaßnahmen

O.Sess_1 Das Sessionhandling SOLL mittels sicherer Frameworks (vgl. O.Ntwk_3) realisiert werden.

O.Sess_2 Die Erstellung von Session-Identifiern MUSS durch den Zufallszahlengenerator des Backends erfolgen.

O.Sess_3 Session-Identifier MÜSSEN als sensitive Daten geschützt werden.

O.Sess_4	Session-Identifier DÜRFEN NICHT unverschlüsselt auf permanenten Speichermedien abgelegt werden.
O.Sess_5	Die Anwendung MUSS die Anwendungssitzung nach einem angemessenen Session-Timeout, gemäß aktueller Best-Practice-Empfehlungen, aktiv beenden.
O.Sess_6	Wird eine Anwendungssitzung beendet, MUSS die Anwendung den Session-Identifier, sowohl auf dem Endgerät, als auch auf dem Backend, sicher löschen.

3.1.6.3 Zustandslose Authentifizierungsmaßnahmen

O.Tokn_1	Das Authentifizierungstoken SOLL auf dem Endgerät in einem sicheren Speicherbereich liegen (z. B. KeyChain/KeyStore). Das Authentifizierungstoken MUSS auf dem Gerät vor einfachem Zugriff durch Dritte geschützt werden (etwa bei rooted/jailbroken Devices).
O.Tokn_2	Es DÜRFEN KEINE sensiblen Daten in ein Authentifizierungstoken eingebettet werden.
O.Tokn_3	Ein Authentifizierungstoken MUSS den voll qualifizierten Namen des Backends umfassen. Die Anwendung MUSS den voll qualifizierten Namen prüfen.
O.Tokn_4	Das Backend MUSS das Authentifizierungstoken mit einem geeigneten Verfahren (vgl. [TR02102-1]) signieren.
O.Tokn_5	Der private Schlüssel, zum Signieren des Authentifizierungstokens DARF NICHT auf dem Gerät vorliegen.
O.Tokn_6	Das Backend MUSS das Token prüfen. Der Signaturtyp DARF NICHT none sein.
O.Tokn_7	Das Backend MUSS Anfragen mit ungültigen oder unsignierten Authentifizierungstoken ablehnen.
O.Tokn_8	Das Backend MUSS eine angemessene Time-To-Live bei der Bewertung der Gültigkeit eines Tokens berücksichtigen.
O.Tokn_9	Das Backend MUSS dem Nutzer bestehende Authentifizierungstoken, auf Anfrage, zur Verfügung stellen.
O.Tokn_10	Das Backend MUSS es dem Nutzer ermöglichen ein oder alle zuvor ausgestellten Authentifizierungstoken ungültig zu machen (etwa bei Geräteverlust).

3.1.7 Prüfaspekt (7): Datenspeicherung und Datenschutz

O.Data_1	Die Werkseinstellung der Anwendung MUSS den maximalen Datenschutz und die maximale Sicherheit bieten.
O.Data_2	Alle sensiblen Daten MÜSSEN verschlüsselt gespeichert werden. Dies gilt sowohl für flüchtiges Ablegen (z. B. im Arbeitsspeicher), als auch für dauerhaftes Speichern

	(z. B. in einer Cloud-Umgebung). Eingeschlossen sind kryptographische Schlüssel, mit Ausnahme der Speicherverschlüsselung. Eine hardwareunterstützte Schlüsselverwaltung der Plattform SOLL bevorzugt verwendet werden. Ist ein hinreichender Schutz der Schlüssel durch die Plattform sichergestellt (z. B. in embedded Secure Element/Trusted Execution Environment), KANN die Applikation Schlüssel im Klartext dort hinterlegen.
O.Data_3	Alle sensiblen Daten SOLLEN in einer vor Einsicht und Manipulation geschützten Umgebung abgelegt werden (z. B. embedded Secure Element/Trusted Execution Environment). Damit SOLL ein für die individuelle Plattform bzw. das individuelle Endgerät, höchstmöglicher Schutz erreicht werden.
O.Data_4	Die Applikation DARF KEINE Ressourcen gegenüber Dritten verfügbar machen, die einen Zugriff auf sensible Daten ermöglichen.
O.Data_5	Alle erhobenen sensiblen Daten DÜRFEN NICHT über die Dauer ihrer jeweiligen Verwendung hinaus in der Anwendung gehalten werden. Hierbei MUSS die Applikation die Grundsätze der Datensparsamkeit und Zweckbindung berücksichtigen.
O.Data_6	Die Speicherung und Verarbeitung von sensiblen Daten SOLL im Backend erfolgen.
O.Data_7	Bei der Verwendung von Aufnahmegeräten (z. B. Kamera) MÜSSEN sämtliche Metadaten mit Datenschutz-Relevanz, wie etwa Rückschlüsse auf den GPS-Koordinaten des Aufnahmeorts, eingesetzte Hardware, etc., entfernt werden.
O.Data_8	Bei der Erhebung von sensiblen Daten, durch die Verwendung von Aufnahmegeräten (z. B. Kamera), MUSS vorgebeugt werden, da andere Anwendungen darauf Zugriff erlangen könnten, etwa über eine Mediengalerie.
O.Data_9	Bei der Eingabe sensibler Daten über die Tastatur SOLL die Anwendung unterbinden, dass Aufzeichnungen für Dritte erkennbar werden. Dies schließt insbesondere Caches, Autokorrektur- und Autovervollständigungsverfahren, Eingabegeräte von Drittanbietern und jegliche für Dritte auswertbare Speicherung, aus.
O.Data_10	Bei der Eingabe sensibler Daten SOLL der Export in die Zwischenablage unterbunden werden. Die Anwendung KANN alternativ eine eigene Zwischenablage implementieren, welche vor dem Zugriff durch andere Apps geschützt ist.
O.Data_11	Sensible Daten wie biometrische Daten oder private Schlüssel, DÜRFEN NICHT aus der Komponente, auf der sie erzeugt wurden, exportiert werden.
O.Data_12	Die Anwendung SOLL beim Anzeigen sensibler Daten den Zugriff für Dritte und die Speicherung des Bildschirms (z. B. Screenshots und Anzeigen für das App-Switching) unterbinden.
O.Data_13	Die Anwendung DARF KEINE sensiblen Daten in Logfiles oder anderen Meldungen oder Benachrichtigungen, die nicht vom Benutzer explizit eingeschaltet wurden (siehe O.Plat_4), schreiben.

O.Data_14	Die Anwendung MUSS sicherstellen, dass im gesperrten Zustand des Endgeräts alle sensiblen Daten verschlüsselt sind. Anwendungshinweis/Beispiel: Ältere Plattformversionen erlauben teilweise die Speicherung der Applikation auf externen Speichermedien, die nicht der Speicherverschlüsselung unterliegen. Dies MUSS die Applikation untersagen, da die obige Anforderung damit nicht erfüllbar wäre.
O.Data_15	Die Applikation MUSS lokal gespeicherte Daten mit einer sicheren Gerätebindung versehen.
O.Data_16	Schützt die Plattform das gewählte Speichermedium nicht gegen Diebstahl (z. B. unverschlüsselte SD-Karten), MUSS die Applikation bei einer Auswahl des betreffenden Speichermediums den Nutzer auf das erhöhte Risiko hinweisen. Anwendungshinweis/Beispiel: Die Verschlüsselung von Daten auf Anwendungsebene bleibt in jedem Fall erhalten.
O.Data_17	Die Anwendung MUSS sicherstellen, dass bei ihrer Deinstallation alle sensiblen Daten und anwendungsspezifischen Anmeldeinformationen auf dem Endgerät vollständig gelöscht werden.
O.Data_18	Die Anwendung MUSS dem Nutzer die Möglichkeit geben, dass bei ihrer Deinstallation alle sensiblen Daten und anwendungsspezifischen Anmeldeinformationen auch im Backend vollständig gelöscht werden. Entscheidet sich der Nutzer die Daten im Backend nicht zu löschen, MUSS eine für den Zweck angemessene maximale Verweildauer definiert sein. Der Nutzer MUSS über die Verweildauer informiert werden. Nach Ablauf der maximalen Verweildauer MÜSSEN alle sensiblen Daten und anwendungsspezifischen Anmeldeinformationen vollständig gelöscht werden. Dem Nutzer MUSS die Möglichkeit gegeben werden alle Daten auch vor Ablauf der Verweildauer vollständig zu löschen. Anwendungshinweis/Beispiel: Ermöglichen von Deinstallation auf Grund von Gerätewechsel ohne Verlust der Historie.
O.Data_19	Um dem Missbrauch von Daten nach einem Geräteverlust entgegenzuwirken, KANN die Anwendung einen Kill-Switch realisieren, d. h. ein absichtliches, sicheres Überschreiben von Nutzerdaten im Gerät auf Applikationsebene, ausgelöst durch das Backend. Der Hersteller MUSS die Auslösung des Kill-Switches durch den Anwender über das Backend durch starke Authentifizierungsmechanismen vor missbräuchlicher Nutzung schützen.

3.1.8 Prüfaspekt (8): Kostenpflichtige Ressourcen

O.Paid_1	Die Applikation MUSS für den Nutzer kenntlich machen, welche Leistungen mit zusätzlichen Kosten vergütet werden.
O.Paid_2	Die Anwendung MUSS vor dem Ausführen kostenpflichtiger Aktionen das Einverständnis des Nutzers einholen.

O.Paid_3	Die Anwendung MUSS, vor einer Zugriffsanforderung (z. B. Android-Berechtigungen) auf kostenpflichtige Ressourcen, das Einverständnis des Nutzers einholen. Anwendungshinweis/Beispiel: Das Versenden von SMS kann Kosten verursachen und benötigt daher ein Einverständnis.
O.Paid_4	Die Anwendung KANN, für den Zugriff auf häufig verwendete, kostenpflichtige Ressourcen, ein dauerhaftes Einverständnis des Nutzers einholen, soweit dies dem Zweck der Anwendung angemessen ist.
O.Paid_5	Die Anwendung MUSS den Nutzer in die Lage versetzen zuvor erteilte Einverständnisse zurückzuziehen.
O.Paid_6	Die Anwendung SOLL die Transaktionshistorie von zahlungspflichtigen Ressourcen und Funktionen im Backend ablegen. Die Transaktionshistorie, einschließlich der Metadaten, MUSS als sensibles Datum gemäß O.Zweck_8 behandelt werden.
O.Paid_7	Falls die Anwendung kostenpflichtige Funktionen anbietet, MUSS der Hersteller ein Konzept vorlegen, welches vorbeugt, dass Dritte die Zahlungsströme zur Nutzung von Anwendungsfunktionen zurückverfolgen können. Anwendungshinweis/Beispiel: Eine periodische Zahlungsabwicklung ist ein möglicher Ansatz, die tatsächliche Nutzungsintensität gegenüber Dritten zu verbergen.
O.Paid_8	Die Anwendung MUSS dem Nutzer eine Übersicht der entstandenen Kosten anbieten. Falls die Kosten aufgrund einzelner Zugriffe erfolgt sind, MUSS die Anwendung einen Überblick der Zugriffe aufführen.
O.Paid_9	Die Validierung von getätigten Bezahlvorgängen MUSS im Backend vorgenommen werden.
O.Paid_10	Zahlverfahren von Drittanbietern MÜSSEN die Anforderungen an Drittanbieter-Software erfüllen (vgl. Kapitel 3.1.4).

3.1.9 Prüfaspekt (9): Netzwerkkommunikation

O.Ntwk_1	Jegliche Netzwerkkommunikation der Anwendung MUSS durchgängig mit TLS verschlüsselt werden.
O.Ntwk_2	Die Konfiguration der TLS-Verbindungen MUSS dem aktuellen Stand der Technik entsprechen und aktuellen Best-Practice-Empfehlungen folgen (vgl. [TR02102-2]).
O.Ntwk_3	Die Anwendung MUSS entweder die Sicherheitsfunktionalität der jeweilig verwendeten Betriebssystem-Plattform oder sicherheitsüberprüfte Frameworks oder Bibliotheken verwenden, um sichere Kommunikationskanäle aufzubauen.

O.Ntwk_4	Die Applikation MUSS Zertifikatspinning unterstützen, d. h. sie DARF KEINE Zertifikate akzeptieren deren Zertifikatskette dem Hersteller nicht vertrauenswürdig erscheint [RFC7469].
O.Ntwk_5	Die Applikation MUSS das Server-Zertifikat des Backends überprüfen.
O.Ntwk_6	Das Backend MUSS Verbindungen, deren Protokollversion oder Cipher Suite die [TR02102-2] nicht erfüllen, ablehnen.
O.Ntwk_7	Die Applikation MUSS die Integrität der Antworten des Backends validieren.
O.Ntwk_8	Die Applikation MUSS plattformspezifische Fallback-Mechanismen (z. B. clear Text traffic Opt-out bzw. analog In-App Transport Security) deaktivieren.
O.Ntwk_9	Die Anwendung SOLL, für alle aufgebauten Verbindungen, Log-Dateien auf dem Backend vorhalten. Dabei MUSS sichergestellt werden, dass bei der Verwendung von intermediären Proxy-Servern die HTTP-Header vollständig miterfasst werden (z. B. X-Forwarded-for).
O.Ntwk_10	Ein abgebrochener Start MUSS als Sicherheitsereignis im Backend protokolliert werden.

3.1.10 Prüfaspekt (10): Plattformspezifische Interaktionen

O.Plat_1	Für die Nutzung der Anwendung MUSS das Endgerät über einen Geräteschutz (Passwort, Mustersperre, o. ä.) verfügen. Der Hersteller MUSS den Nutzer über die Folgen eines nicht aktivierten Geräteschutzes aufklären.
O.Plat_2	Die Anwendung DARF NUR die Berechtigungen einfordern, die für die Erfüllung ihres primären Zwecks notwendig sind.
O.Plat_3	Die Applikation MUSS den Nutzer auf den Zweck der anzufragenden Berechtigungen und auf die Auswirkungen hinweisen, die eintreten, falls der Nutzer diese nicht gewährt.
O.Plat_4	Die Applikation KANN dem Nutzer die Optionen bieten, Meldungen und Benachrichtigungen, ggf. auch mit sensiblen Inhalten, anzuzeigen. Bei Werkseinstellung MUSS diese deaktiviert sein.
O.Plat_5	Die Applikation SOLL den Zugriff auf vorgesehene Dateipfade beschränken. Anwendungshinweis/Beispiel: Zum Beispiel durch Whitelisting von absoluten Dateipfaden.
O.Plat_6	Die Anwendung MUSS Zugriffsbeschränkungen auf sämtliche Daten realisieren.
O.Plat_7	Die Applikation MUSS Broadcast-Nachrichten auf autorisierte Applikationen beschränken.
O.Plat_8	Die Anwendung DARF in Broadcast-Nachrichten KEINE sensiblen Daten versenden.

O.Plat_9	Das Anbieten von sensiblen Funktionalitäten über Interprozesskommunikation SOLL unterbunden werden. Ist das Anbieten zur Erfüllung des Zwecks erforderlich, MÜSSEN die angebotenen Funktionalitäten angemessen geschützt werden.
O.Plat_10	Die Applikation SOLL verhindern, dass JavaScript während der Nutzung von WebView aktiv ist. Falls JavaScript für die Realisierung der Anwendung unabdingbar ist, MUSS die Applikation JavaScript aus Quellen außerhalb der Kontrolle des Herstellers ablehnen.
O.Plat_11	Wechselt die Anwendung in den Hintergrundmodus, MUSS diese alle sensiblen Daten aus der aktuellen Ansicht (Views in iOS bzw. Activities in Android) entfernen.
O.Plat_12	Die Applikation MUSS alle nicht benötigten Protokoll-Handler in WebViews deaktivieren.
O.Plat_13	Die Applikation MUSS nach Beenden der Anwendung anwendungsspezifische Cookies löschen.
O.Plat_14	Die Applikation SOLL nach Beenden alle nutzerspezifischen Daten im Arbeitsspeicher sicher überschreiben.

3.1.11 Prüfaspekt (11): Resilienz

O.Resi_1	Die Anwendung MUSS dem Nutzer barrierearme Best-Practice-Empfehlungen, zum sicheren Umgang mit der Anwendung und ihrer Konfiguration bereitstellen.
O.Resi_2	Die Anwendung MUSS gerootete oder gejailbreakte Geräte entsprechend dem aktuellen Stand der Technik erkennen und angemessen darauf reagieren. Der Hersteller MUSS darstellen, welche Risiken für die Daten des Nutzers bei einer Fortsetzung der App bestehen (z. B. dass diese offengelegt werden könnten). Eine weitere angemessene Reaktion ist die Beendigung der Anwendung.
O.Resi_3	Die Anwendung MUSS den Start in einer Entwicklungs-/Debugumgebung sicher erkennen und unterbinden.
O.Resi_4	Die Anwendung MUSS ihren Start abbrechen, falls sie unter ungewöhnlichen Benutzerrechten gestartet wird (z. B. root oder nobody).
O.Resi_5	Die Anwendung MUSS die Integrität des Endgeräts überprüfen, bevor sensible Daten verarbeitet werden (z. B. Google Safety.Net).
O.Resi_6	Die Anwendung MUSS vor dem Zugriff auf das Backend dessen Integrität überprüfen (siehe auch O.Ntwk_4).
O.Resi_7	Die Applikation SOLL Härtungsmaßnahmen, wie etwa eine Integritätsprüfung vor jeder Verarbeitung sensibler Daten innerhalb des Programmablaufs, realisieren.

O.Resi_8	Die Applikation MUSS starke Maßnahmen gegen Reverse Engineering umsetzen und KANN dafür auf Verschleierungsmaßnahmen, wie Code-Obfuskierung und Stringverschlüsselung, zurückgreifen.
O.Resi_9	Die Anwendung MUSS Zugriffskontrollmechanismen, unter der Berücksichtigung von unterschiedlichen Plattformen und Plattformversionen implementieren, so dass ein missbräuchlicher Zugriff auf Ressourcen, durch eine Änderung der Plattformversion, ausgeschlossen ist und somit in jeder Ausführungsumgebung ein hinreichender Schutz aller Assets erzielt wird.

Literaturverzeichnis

[ASVS] The OWASP Foundation, „Application Security Verification Standard“, Version 4.0, verfügbar unter https://www.owasp.org/images/d/d4/OWASP_Application_Security_Verification_Standard_4.0-en.pdf

[GDR18] we are social, „Global Digital Report 2018“, Version Januar 2018, verfügbar unter https://wearesocial.com/de/blog/2018/01/global-digital-report-2018

[iOSSF] Apple Inc. „iOS Security Framework“, verfügbar unter https://developer.apple.com/documentation/security

[KCC-C5] Bundesamt für Sicherheit in der Informationstechnik, „Kriterienkatalog Cloud Computing“, Version 2020, verfügbar unter https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/CloudComputing/Anforderungskatalog/2020/C5_2020.pdf?__blob=publicationFile&v=2

[MASVS] The OWASP Foundation, „ Mobile AppSec Verification Standard (German Translation)“, Version 1.1, verfügbar unter https://github.com/OWASP/owasp-masvs/releases/download/1.1.4/OWASP_Mobile_AppSec_Verification_Standard_1.1.4_Document-de.pdf

[MSTG] The OWASP Foundation, „Mobile Security Testing Guide“, Version December 2016, verfügbar unter https://www.enisa.europa.eu/publications/smartphone-secure-development-guidelines-2016/at_download/fullReport

[NSP80057] National Institute of Standards and Technology „Recommendation for Key Management“, Revision 4, verfügbar unter https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf

[RFC7469] C. Evans, C. Palmer, R. Sleevi, Google Inc., „Public Key Pinning Extension for HTTP“, Version April 2015, verfügbar unter https://tools.ietf.org/html/rfc7469

[SfAD] Google Developers, „Security for Android Developers“, Version Januar 2020, verfügbar unter https://developer.android.com/topic/security

[SSDG] European Union Agency For Network And Information Security, „Smartphone Secure Development Guidelines“, Version December 2016, verfügbar unter https://www.enisa.europa.eu/publications/smartphone-secure-development-guidelines-2016/at_download/fullReport

[T10WASR] The OWASP Foundation, “Top 10 -2017”, Version 2017, verfügbar unter https://www.owasp.org/images/9/90/OWASP_Top_10-2017_de_V1.0.pdf

[TR02102-1] Bundesamt für Sicherheit in der Informationstechnik, “Kryptographische Verfahren: Empfehlungen und Schlüssellängen”, Version 2019-01, verfügbar unter https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf?__blob=publicationFile&v=10

[TR02102-2] Bundesamt für Sicherheit in der Informationstechnik, “Kryptographische Verfahren: Empfehlungen und Schlüssellängen Teil 2 – Verwendung von Transport Layer Security (TLS)”, Version 2019-01, verfügbar unter https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102-2.pdf?__blob=publicationFile&v=7

[SGBV33a] Bundesanzeiger, “Sozialgesetzbuch (SGB) Fünftes Buch (V) - Gesetzliche Krankenversicherung - § 33a Digitale Gesundheitsanwendungen”, Version Dezember

2019, verfügbar unter http://www.bgbl.de/xaver/bgbl/start.xav?
startbk=Bundesanzeiger_BGBl&jumpTo=bgbl119s2562.pdf

Abkürzungsverzeichnis

API	Application Programming Interface (Anwendungs-/Programmierschnittstelle)
App	Applikation
A.*	Assumption (Annahme)
BSI	Bundesamt für Sicherheit in der Informationstechnik
GPS	Global Positioning System
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IoT	Internet of Things
O.*	Objective (Prüfaspekt)
OSP.*	Organizational Security Policies (Organisatorische Sicherheitspolitiken)
SD-Karte	Secure Digital Memory Card
SDK	Software Development Kit
SGB V	Sozialgesetzbuch (SGB) Fünftes Buch (V)
SMS	Short Message Service
SPD	Security Problem Definition
SSID	Service Set Identifier
T.*	Threat (Bedrohung)
TR	Technische Richtlinie
TLS	Transport Layer Security
URL	Uniform Resource Locator
WiFi	Wireless Fidelity
WLAN	Wireless Local Area Network
XML	Extensible Markup Language